Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №1 по курсу «Моделирование»

Тема Решение задачи Коши методами Пикара, Эйлера и Рунге-Кутта

Студент Кононенко С.С.

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва — 2021 г.

Тема работы

Программная реализация приближенного аналитического метода и численных алгоритмов первого и второго порядков точности при решении задачи Коши для ОДУ.

Цель работы

Получение навыков решения задачи Коши при помощи методов Пикара, Эйлера и Рунге-Кутта 2-го порядка.

Теоретические сведения

Имеем ОДУ, у которого отсутствует аналитическое решение:

$$\begin{cases} u'(x) = f(x, u) \\ u(\xi) = \eta \end{cases} \tag{1}$$

Для решения данного ОДУ были использованы 3 алгоритма.

Метод Пикара

Имеем:

$$u(x) = \eta + \int_{\xi}^{x} f(t, u(t))\, dt \tag{2}$$

Строим ряд функций:

$$y^{(s)} = \eta + \int_{\xi}^{x} f(t, y^{(s-1)}(t))\, dt, \qquad y^{(0)} = \eta \tag{3}$$

Построим 4 приближения для уравнения (2):

$$y^{(1)}(x) = 0 + \int_0^x t^2\,dt = \frac{x^3}{3} \tag{4}$$

$$y^{(2)}(x) = 0 + \int_0^x (t^2 + \left(\frac{t^3}{3}\right)^2)\,dt = \frac{x^3}{3} + \frac{x^7}{63} \tag{5}$$

$$y^{(3)}(x) = 0 + \int_0^x (t^2 + \left(\frac{t^3}{3} + \frac{t^7}{63}\right)^2)\,dt = \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535} \tag{6}$$

$$\begin{aligned} y^{(4)}(x) = 0 + \int_0^x (t^2 + \left(\frac{t^3}{3} + \frac{t^7}{63} + \frac{2t^{11}}{2079} + \frac{t^{15}}{59535}\right)^2)\,dt = \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \\ \frac{x^{15}}{59535} + \frac{2x^{15}}{93555} + \frac{2x^{19}}{3393495} + \frac{2x^{19}}{2488563} + \frac{2x^{23}}{86266215} + \\ \frac{x^{23}}{99411543} + \frac{2x^{27}}{3341878155} + \frac{x^{31}}{109876902975} \end{aligned} \tag{7}$$

Метод Эйлера

$$y^{(n+1)}(x) = y^{(n)}(x) + h \cdot f(x_n, y^{(n)}) \tag{8}$$

Порядок точности: $O(h)$.

Метод Рунге-Кутта

$$y^{n+1}(x) = y^n(x) + h((1-\alpha)R_1 + \alpha R_2) \tag{9}$$

где $R_1 = f(x_n, y^n)$, $R_2 = f(x_n + \frac{h}{2\alpha}, y^n + \frac{h}{2\alpha}R_1)$, $\alpha = \frac{1}{2}$ или 1

Порядок точности: $O(h^2)$.

Исходный код алгоритмов

В листингах 1 – 3 представлены реализации алгоритмов Пикара, Эйлера и Рунге-Кутта соответственно. В листингах 4 – 7 приведены вспомогательные функции и главная программа.

Листинг 1 – Реализация алгоритма Пикара

```
package cauchy

import "math"

func fapprox(x float64) float64 {
    return x * x * x / 3
}

func sapprox(x float64) float64 {
    return fapprox(x) + math.Pow(x, 7)/63
}

func tapprox(x float64) float64 {
    return sapprox(x) + 2*math.Pow(x, 11)/2079 + math.Pow(x, 15)/59535
}

func foapprox(x float64) float64 {
    return sapprox(x) + 2*math.Pow(x, 11)/2079 + 13*math.Pow(x, 15)/218295 +
        82*math.Pow(x, 19)/37328445 + 662*math.Pow(x, 23)/10438212015 +
        4*math.Pow(x, 27)/3341878155 + math.Pow(x, 31)/109876902975
}

// Picard used to solve Cauchy problem with Picard method
func Picard(x0, h float64, n int) FMat64 {
    r := MakeFMat64(4, 0)

    for i := 0; i <= n; i++ {
        r[0] = append(r[0], fapprox(x0))
        r[1] = append(r[1], sapprox(x0))
        r[2] = append(r[2], tapprox(x0))
        r[3] = append(r[3], foapprox(x0))

        x0 += h
    }

    return r
```

```
}
```

Листинг 2 – Реализация алгоритма Эйлера

```
package cauchy

// Euler used to solve Cauchy problem with Euler method
func Euler(x0, y0, h float64, n int) FArr64 {
    r := make(FArr64, 0)

    for i := 0; i <= n; i++ {
        r = append(r, y0)
        y0 += h * equation(x0, y0)
        x0 += h
    }

    return r
}
```

Листинг 3 – Реализация алгоритма Рунге-Кутта

```
package cauchy

// RungeKutta used to solve Cauchy problem with Runge-Kutta method
func RungeKutta(x0, y0, a, h float64, n int) FArr64 {
    r := make(FArr64, 0)

    for i := 0; i <= n; i++ {
        r = append(r, y0)
        y0 += h * ((1-a)*equation(x0, y0) + a*equation(x0+h/2/a, y0+h*equation(x0,
            y0)/2/a))
        x0 += h
    }

    return r
}
```

Листинг 4 – Исходное уравнение

```
package cauchy

func equation(x, u float64) float64 {
    return x*x + u*u
}
```

Листинг 5 – Реализация вспомогательных типов

```go
package cauchy

// FArr64 used to represent []float64
type FArr64 []float64

// FMat64 used to represent [][]float64
type FMat64 []FArr64

// MakeFMat64 used to initialize FMat64
func MakeFMat64(n, m int) FMat64 {
    mat := make(FMat64, n)
    for i := 0; i < n; i++ {
        mat[i] = make(FArr64, m)
    }

    return mat
}
```

Листинг 6 – Реализация табличного вывода

```go
package cauchy

import (
    "fmt"
    "strings"

    "github.com/logrusorgru/aurora/v3"
)

// Log used
func Log(p FMat64, e, rk, x FArr64) {
    fmt.Printf("%200v\n", aurora.BgRed("CAUCHY␣PROBLEM␣SOLUTION"))
    fmt.Printf("%v%198v%v\n", "+", strings.Repeat("-", 198), "+")
    fmt.Printf(
        "|%27v|%27v|%27v|%27v|%28v|%28v|%28v|\n",
        aurora.Green("X"),
        aurora.Green("Picard,␣1st␣approx."),
        aurora.Green("Picard,␣2nd␣approx."),
        aurora.Green("Picard,␣3rd␣approx."),
        aurora.Green("Picard,␣4th␣approx."),
        aurora.Green("Euler"),
        aurora.Green("Runge-Kutta"),
    )
    fmt.Printf("%v%198v%v\n", "+", strings.Repeat("-", 198), "+")
    for i := 0; i < len(e); i += 500 {
        fmt.Printf(
            "|%27.5f|%27.5f|%27.5f|%27.5f|%28.5f|%28.5f|%28.5f|\n",
```

```
            x[i],
            p[0][i],
            p[1][i],
            p[2][i],
            p[3][i],
            e[i],
            rk[i],
        )
    }
    fmt.Printf("%v%36v%v\n", "+", strings.Repeat("-", 198), "+")
}
```

Листинг 7 – Главная программа

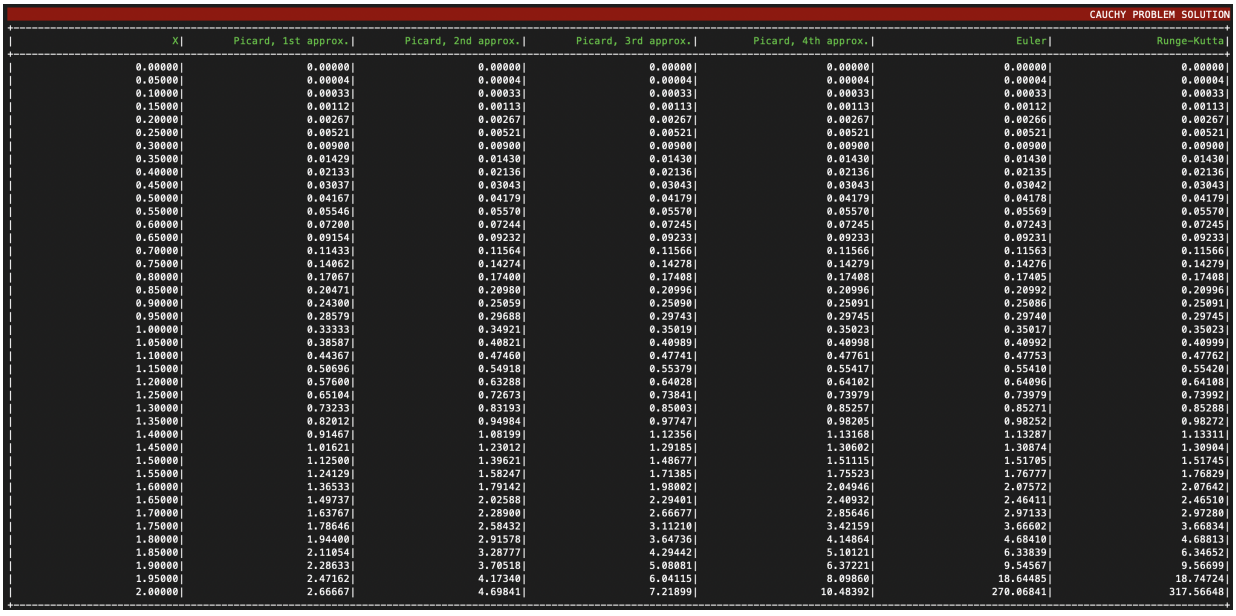
```
package main

import (
    "fmt"
    "math"
    "src/cauchy"
)

func main() {
    xs := 0.
    xe := 2.
    ys := 0.
    h := 1e-4
    n := int(math.Ceil(math.Abs(xe-xs) / h))

    x := make(cauchy.FArr64, 0)
    xn := xs
    for i := 0; i <= n; i++ {
        x = append(x, xn)
        xn += h
    }
    picardSol := cauchy.Picard(xs, h, n)
    eulerSol := cauchy.Euler(xs, ys, h, n)
    rungeKuttaSol := cauchy.RungeKutta(xs, ys, 0.5, h, n)

    fmt.Println()
    cauchy.Log(picardSol, eulerSol, rungeKuttaSol, x)
}
```

Результат работы программы

На рисунке 1 представлен пример работы программы с данными:

- $x_0 = 0$
- $x_n = 2$
- $y_0 = 0$
- $h = 10^{-4}$
- $\alpha = 0.5$

Шаг вывода - 0.05.



CAUCHY PROBLEM SOLUTION

X	Picard, 1st approx.	Picard, 2nd approx.	Picard, 3rd approx.	Picard, 4th approx.	Euler	Runge-Kutta
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.05000	0.00004	0.00004	0.00004	0.00004	0.00004	0.00004
0.10000	0.00033	0.00033	0.00033	0.00033	0.00033	0.00033
0.15000	0.00112	0.00113	0.00113	0.00113	0.00112	0.00113
0.20000	0.00267	0.00267	0.00267	0.00267	0.00266	0.00267
0.25000	0.00521	0.00521	0.00521	0.00521	0.00521	0.00521
0.30000	0.00900	0.00900	0.00900	0.00900	0.00900	0.00900
0.35000	0.01429	0.01430	0.01430	0.01430	0.01430	0.01430
0.40000	0.02133	0.02136	0.02136	0.02136	0.02135	0.02136
0.45000	0.03037	0.03043	0.03043	0.03043	0.03042	0.03043
0.50000	0.04167	0.04179	0.04179	0.04179	0.04178	0.04179
0.55000	0.05546	0.05570	0.05570	0.05570	0.05569	0.05570
0.60000	0.07200	0.07244	0.07245	0.07245	0.07243	0.07245
0.65000	0.09154	0.09232	0.09233	0.09233	0.09231	0.09233
0.70000	0.11433	0.11564	0.11566	0.11566	0.11563	0.11566
0.75000	0.14062	0.14274	0.14278	0.14279	0.14276	0.14279
0.80000	0.17067	0.17400	0.17408	0.17408	0.17405	0.17408
0.85000	0.20471	0.20980	0.20996	0.20996	0.20992	0.20996
0.90000	0.24300	0.25059	0.25090	0.25091	0.25086	0.25091
0.95000	0.28579	0.29688	0.29743	0.29745	0.29740	0.29745
1.00000	0.33333	0.34921	0.35019	0.35023	0.35017	0.35023
1.05000	0.38587	0.40821	0.40989	0.40998	0.40992	0.40999
1.10000	0.44367	0.47460	0.47741	0.47761	0.47753	0.47762
1.15000	0.50696	0.54918	0.55379	0.55417	0.55410	0.55420
1.20000	0.57600	0.63288	0.64028	0.64102	0.64096	0.64108
1.25000	0.65104	0.72673	0.73841	0.73979	0.73979	0.73992
1.30000	0.73233	0.83193	0.85003	0.85257	0.85271	0.85288
1.35000	0.82012	0.94984	0.97747	0.98205	0.98252	0.98272
1.40000	0.91467	1.08199	1.12356	1.13168	1.13287	1.13311
1.45000	1.01621	1.23012	1.29185	1.30602	1.30874	1.30904
1.50000	1.12500	1.39621	1.48677	1.51115	1.51705	1.51745
1.55000	1.24129	1.58247	1.71385	1.75523	1.76777	1.76829
1.60000	1.36533	1.79142	1.98002	2.04946	2.07572	2.07642
1.65000	1.49737	2.02588	2.29401	2.40932	2.46411	2.46510
1.70000	1.63767	2.28900	2.66677	2.85646	2.97133	2.97280
1.75000	1.78646	2.58432	3.11210	3.42159	3.66602	3.66834
1.80000	1.94400	2.91578	3.64736	4.14864	4.68410	4.68813
1.85000	2.11054	3.28777	4.29442	5.10121	6.33839	6.34652
1.90000	2.28633	3.70518	5.08081	6.37221	9.54567	9.56699
1.95000	2.47162	4.17340	6.04115	8.09860	18.64485	18.74724
2.00000	2.66667	4.69841	7.21899	10.48392	270.06841	317.56648

Рисунок 1 – Демонстрация работы программы

- Первый столбец – значение x
- Второй столбец – метод Пикара (1 порядок)
- Третий столбец – метод Пикара (2 порядок)
- Четвертый столбец – метод Пикара (3 порядок)
- Пятый столбец – метод Пикара (4 порядок)

- Шестой столбец – метод Эйлера
- Седьмой столбец – метод Рунге-Кутта

Ответы на контрольные вопросы

Вопрос 1. Укажите интервалы значений аргумента, в которых можно считать решением заданного уравнения каждое из первых 4-х приближений Пикара. Точность результата оценивать до второй цифры после запятой. Объяснить свой ответ.

Ответ. Первое приближение Пикара можно считать решением уравнения до тех пор, пока совпадают результаты для первого и второго приближений до второго знака после запятой. Второе приближение Пикара можно считать решением уравнения до тех пор, пока совпадают результаты для второго и третьего приближений до второго знака после запятой. Третье приближение Пикара можно считать решением уравнения до тех пор, пока совпадают результаты для третьего и четвертого приближений до второго знака после запятой. Четвертое приближение Пикара можно считать решением уравнения до тех пор, пока совпадают результаты для четвертого и пятого приближений до второго знака после запятой.

Основываясь на таблице резлуьтатов:

- 1-ое приближение можно сичтать решением на отрезке $[0, 0.85]$;
- 2-ое приближение можно сичтать решением на луче $(0.85, 1]$;
- 3-е приближение можно сичтать решением на луче $(1, 1.35]$;
- Для определения промежутка, на котором 4-ое приближение является решением, необходимо найти 5-ое приближение, что не выполнялось в данной работе.

Вопрос 2. Пояснить, каким образом можно доказать правильность полученного результата при фиксированном значении аргумента в численных методах.

Ответ. Для доказательства правильности полученного результата необходимо рассмотреть значения на небольшом интервале, располагающемся в окрестности точки, с небольшим шагом (шаг должен стремиться к нулю). Если при шаге, приближенному к нулю, полученные в окрестности точки значения примерно совпадают, то расчет значений был произведен правильно.

Вопрос 3. Каково значение функции при $x = 2$, т.е. привести значение $U(2)$.

Ответ. Примерно 317.6